Measuring the Similarity of Geometric Graphs

Otfried Cheong[1,⋆], Joachim Gudmundsson[2], Hyo-Sil Kim[1,⋆],
Daria Schymura[3,⋆⋆], and Fabian Stehn[3,⋆⋆]

[1] KAIST, Korea
{otfried,hyosil}@tclab.kaist.ac.kr
[2] NICTA,⋆⋆⋆ Australia
joachim.gudmundsson@nicta.com.au
[3] FU Berlin, Germany
{schymura,stehn}@inf.fu-berlin.de

Abstract. What does it mean for two geometric graphs to be similar? We propose a distance for geometric graphs that we show to be a metric, and that can be computed by solving an integer linear program. We also present experiments using a heuristic distance function.

1 Introduction

Computational geometry has studied the matching and analysis of geometric shapes from a theoretical perspective and developed efficient algorithms measuring the *similarity* of geometric objects. Two objects are similar if they do not differ much geometrically. A survey by Alt and Guibas [1] describes the significant body of results obtained by researchers in computational geometry in this area.

This paradigm fits a number of practical shape matching problems quite well, such as the recognition of symmetries in molecules, or the self-alignment of a satellite based on star patterns. Other pattern recognition problems, however, seem to require a different definition of "matching." For instance, recognizing logos, Egyptian hieroglyphics, Chinese characters, or electronic components in a circuit diagram are typical examples where this is the case. The same "pattern" can appear in a variety of shapes that differ geometrically. What remains invariant, however, is the "combinatorial" structure of the pattern.

We propose to consider such patterns as geometric graphs, that is, planar graphs embedded into the plane with straight edges. Two geometric graphs can

⋆ This work was supported by the Korea Science & Engineering Foundation through the Joint Research Program (F01-2006-000-10257-0).

⋆⋆ This research was partially funded by Deutsche Forschungsgemeinschaft within the Research Training Group (Graduiertenkolleg) "Methods for Discrete Structures", the DFG-projects AL253/6-1, KN591/2-2 and through the Joint Research Project 446 KOR 113/211/0-1.

⋆⋆⋆ NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

J. Vahrenhold (Ed.): SEA 2009, LNCS 5526, pp. 101–112, 2009.

be considered similar if both the underlying graph and the geometry of the planar embedding are "similar." The distance measures considered in computational geometry, such as the Hausdorff distance, Fréchet distance, or the symmetric difference, do not seem to apply to geometric graphs.

Pattern recognition systems that combine a combinatorial component with a geometric component are already used in practice—in fact, *syntactic* or *structural* pattern recognition is based on exactly this idea: A syntactic recognizer decomposes the pattern into geometric primitives and makes conclusions based on the appearance and relative position of these primitives [2,7]. While attractive from a theoretical point of view, syntactic recognizers have not been able to compete with numerical or AI techniques for character recognition [6]. In general, the pattern recognition community may be said to consider graph representations as expressive, but too time-consuming, as subgraph isomorphism in general is known to be intractable.

An established measure of similarity between (labeled) graphs is the *edit distance*. The idea of an edit distance is very intuitive: To measure the difference between two objects, measure how much one object has to be changed to be transformed into the other object. To define an edit distance, one therefore defines a set of allowed operations, each associated with a cost. An edit sequence from object A to object B is a finite sequence of allowed operations that transforms A into B. The distance between A and B is the minimal cost of an edit sequence from A to B.

The edit distance originally stems from string matching where the allowed operations are insertion, deletion and substitution of characters. The edit distance of strings can be computed efficiently, and the string edit distance is used widely, for instance in computational biology.

Justice and Hero [5] defined an edit distance for vertex-labeled graphs that additionally allows relabeling of vertices, and give an integer linear programming formulation of the edit distance. The edit operations are insertion and deletion of vertices, insertion and deletion of edges, and a change of a vertex label.

It is natural to try to define an edit distance for geometric graphs as well. Simply considering a geometric graph as a graph whose vertices are labeled with their coordinates is not sufficient, as the cost of inserting and deleting an edge should also be dependent on the length of the edge. This leads to the following operations: Insertions and deletions of vertices, translations of vertices, and insertions and deletions of edges. However, it is difficult to give bounds on the length of an edit sequence: vertices can move several times to make insertions and deletions cheaper. We give some examples in the following section.

This leads us to define another graph distance function in Section 3. It is not an edit distance, and so we need to prove explicitly that it is a metric. We also give an integer linear programming formulation that allows us to compute our distance for small graphs with an ILP solver. Unfortunately, we do not know how to compute or even approximate our graph distance for larger graphs. In fact, we give two reductions from NP-hard problems, but both result in non-"practical" instances of the problem.